УДК 004.75 **DOI: https://doi.org/10.30837/PMW.2026.T1.054**

РОЗПОДІЛЕНА МІКРОСЕРВІСНО-ОРІЄНТОВАНА ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ОБРОБКИ АСТРОНОМІЧНИХ ДАНИХ З ВИКОРИСТАННЯМ СПЕЦИФІКАЦІЇ OPENAPI

Хламов С.В.
к.т.н., доцент, кафедра «Медіасистеми та технології»,
Харківський національний університет радіоелектроніки
ORCID ID: 0000-0001-9434-1081
Орлов С.В.
аспірант, кафедра «Медіасистеми та технології»,
Харківський національний університет радіоелектроніки
ORCID ID: 0009-0008-0680-206X
Табакова І.С.
к.т.н., професор, кафедра «Медіасистеми та технології»,
Харківський національний університет радіоелектроніки
ORCID ID: 0000-0001-6629-4927

Анотація. *Розділ присвячено застосуванню специфікації OpenAPI у розподіленій мікросервісно-орієнтованій системі для обробки астрономічних даних. Метою є автоматизація процесів обробки із використанням методів інтелектуального аналізу даних та KDD. Запропонований підхід підвищує інтероперабельність, масштабованість і зручність супроводу системи. Каркас системи реалізовано на платформі .NET Core мовою C# з інтеграцією Swagger. Рішення може застосовуватись для обробки астрономічних зображень, реалізації концепції Virtual Observatory та інтеграції з CI/CD.*

Ключові слова: *інформаційна система, клієнт-серверна архітектура, мікросервісно-орієнтована архітектура, масштабованість, Swagger, OpenAPI, rest API, JSON, .net.*

Вступ

Астероїдно-кометна небезпека стає величезною потенційною проблемою у XXI столітті [1], яка може спричинити глобальні руйнування, зіткнення з геостаціонарними штучними супутниками [2], космічним сміттям тощо. Щоб уникнути такої ситуації, людство постійно розробляє та вдосконалює математичні методи [3] та алгоритми для астрономічного наукового напрямку, такого як обробка астрономічних зображень та комп'ютерний зір [4], що включає вирівнювання фону [5], вирівнювання яскравості [6], астрометричне зменшення [7], фотометричне зменшення [8], виявлення рухомих об'єктів у серії кадрів або навіть відкриття об'єктів Сонячної системи (SSO) [9], таких як комети, астероїди [10], малі планети, галактики, зірки тощо.

Усі астрономічні наукові спостереження створюються за допомогою приладів із зарядовим зв'язком (CCD) [11], які використовуються як основне

обладнання в телескопах або будь-якій іншій оптичній системі в обсерваторіях. Такі астрономічні наукові спостереження збираються протягом заданого періоду спостереження досліджуваних малих небесних SSO [12], а також штучних супутників. Після проведення серії спостережень досліджуваних SSO необхідно проаналізувати результати спостережень, які можуть включати визначення періоду та форми обертань таких досліджуваних SSO. Це означає, що існує багато великих астрономічних даних, і для їх обробки потрібно застосовувати різні підходи інформаційних технологій.

Астрономічна наукова інформація також може бути зібрана з різних історичних кластерів, архівів, віртуальних обсерваторій [13], хмар даних, астрономічних астрометричних та фотометричних каталогів [14], різних серверів та інших сховищ.

Мета та задачі дослідження

Спільною метою всіх науково-технологічних алгоритмів та методів є автоматизація якомога більшої кількості процесів без будь-яких людських дій. У загальних випадках це може бути зроблено за допомогою різних астрономічних наукових інформаційних систем. У цих інформаційних системах для пришвидшення та оптимізації обробки астрономічних даних використовуються різні завдання інтелектуального аналізу даних [15] та пошуку знань у базах даних (KDD) [16].

У випадку, якщо астрономічна наукова інформаційна система є дуже складною та складається з різних математичних модулів та бібліотек, вона стає розподіленою інформаційною системою, орієнтованою на мікросервіси, для обробки астрономічних даних.

Мікросервіси, також відомі як мікросервісна архітектура, – це архітектурний стиль, який структурує застосунок як набір слабо пов'язаних сервісів, кожен з яких реалізує бізнес-можливості. Мікросервісна архітектура забезпечує безперервну доставку та розгортання великих, складних інформаційних систем. Вона також дозволяє організації розвивати свій технологічний стек, масштабувати та бути більш стійкою з часом. Мікросервісна архітектура виступає за перетворення єдиної інформаційної системи на набір слабо пов'язаних сервісів. Ці блоки також забезпечують безперервну доставку та розгортання великих, монолітних інформаційних систем з мінімальною потребою в централізації.

Зі зростанням популярності мікросервісної архітектури [17] зростає складність управління кількома взаємопов'язаними сервісами. Документація стає важливою не лише для зовнішніх користувачів, але й для внутрішніх розробників, яким потрібно розуміти API, що надаються кожним сервісом. Саме тут і вступає в гру Swagger. Swagger, тепер відомий як специфікація OpenAPI, – це потужний інструмент для опису, створення, використання та візуалізації RESTful веб-сервісів.

Swagger спрощує розробку та підтримку API, надаючи інтерфейс, незалежний від мови, для REST API [18]. За допомогою Swagger ви можете створювати клієнтські бібліотеки, серверні заглушки та документацію API, що сприяє чіткій комунікації між вашою командою розробників та за її межами. Це гарантує, що всі мікросервіси розмовляють однією «мовою», коли йдеться про кінцеві точки API, параметри та моделі даних.

Цей розділ спрямований на аналіз основних напрямків та особливостей специфікації OpenAPI для розробки інформаційних систем, орієнтованих на мікросервіси. Реальні приклади астрономічної системи обробки даних реалізовані за допомогою фреймворку .Net Core та мови програмування C#, яка ідеально підходить для розробки розподіленої інформаційної системи, орієнтованої на мікросервіси.

У розділі 2 представлено кілька технологій, пов'язаних з нашою роботою для вирішення завдання документування API. У розділі 3 детально описано архітектуру системи на основі стилю архітектури мікросервісів, представлено інтеграцію специфікації Swagger OpenAPI в реальну реалізацію мікросервісів. У цьому розділі також представлені інтегровані моделі даних для астрономічної системи обробки даних, а також проілюстровано результат виконання. Цей розділ також спрямований на обговорення переваг запропонованого використання специфікації OpenAPI в розподіленій мікросервісно-орієнтованій інформаційній системі для обробки астрономічних даних.

У розділі 4 представлено дискусійну панель з перевагами та недоліками, методами та недоліками запропонованих компонентів розподіленої мікросервісно-орієнтованої інформаційної системи для обробки астрономічних даних та їх аналогів.

Дослідження завершується висновком, який ілюструє основні досягнення досліджень та напрямки майбутньої роботи, а також можливості для додаткових досліджень та вдосконалень.

Огляд літератури

Кожен SSO в цифровому кадрі має типову форму свого зображення [19]. Для виявлення/розпізнавання таких зображень SSO та оцінки їх позиційних та рухових параметрів [22] розроблені загальні методи обробки зображень [20] та машинного зору [21]. Такі методи базуються на аналізі лише тих пікселів, які потенційно належать досліджуваному об'єкту. Недоліками таких методів є дуже низька точність, коли типова форма об'єкта має іншу форму [23].

Методи оцінки апертурної яскравості [24] зображень об'єктів працюватимуть лише з одним зображенням кожного SSO. Будь-які методи узгодженої фільтрації [25, 26] та високочастотної фільтрації [27], які присвячені покращенню якості пошкоджених зображень, є дуже ресурсоємними. Недоліками цих методів є велика складність та низька точність під час обробки астрономічних даних, коли зображення об'єкта має кілька піків величини.

Методи вейвлет-аналізу [28] або навіть аналізу часових рядів [29] не такі ефективні, оскільки ми не маємо великого обсягу вхідних даних для аналізу.

Також недоліком таких алгоритмів є спотворення загальної статистики та можливість обробляти лише чіткі вимірювання без будь-яких відхилень у типовій формі зображення.

Будь-які методи глибокого навчання та розпізнавання образів [30, 31] також вимагають великої кількості астрономічних даних для навчання. Проблема таких методів полягає в тому, що астрономічне зображення має багато артефактів, тому в серії кадрів виявляється багато хибних об'єктів.

У цьому випадку для роботи з ресурсоємними математичними алгоритмами, методами та модулями, які їх реалізують, потрібна розподілена архітектура інформаційної системи, орієнтована на мікросервіси, для обробки астрономічних даних. І специфікації OpenAPI є хорошим підходом для таких цілей.

Існує кілька альтернатив Swagger для реалізації специфікацій OpenAPI, кожна з яких пропонує унікальні функції та переваги, які можуть бути більш підходящими залежно від ваших конкретних вимог. Ось деякі помітні альтернативи, згадані нижче.

Postman – це універсальний інструмент для розробки та тестування API [32]. Postman забезпечує автоматизоване тестування, командну співпрацю та інтеграцію з різними інструментами CI/CD. Він також включає такі функції, як макетні сервери та інтерактивна документація API, що робить його комплексним рішенням для управління життєвим циклом API. Автор описує архітектуру мікросервісів як масштабований метод проектування та впровадження онлайн-додатків. Через свою мережеву природу, мікросервісні додатки потребують тестування в мережевому середовищі.

Автоматизація цих тестів передбачає генерування штучного мережевого трафіку, зазвичай у формі HTTP-запитів до API, таких як REST API. Ці теми досліджуються з точки зору проектування та впровадження тестів, а також ключових особливостей архітектури мікросервісів та автоматизованого тестування загалом. В основі цієї дисертації детально описано процес проектування та впровадження системи автоматизації тестування для Intel Insight, а також платформи автоматичного зберігання зображень та обробки фотограмметрії, побудованої як система мікросервісів.

Платформа Stoplight перевершує інші галузі в галузі проектування, документування та управління API [33]. Вона має зручний інтерфейс для створення специфікацій API за допомогою OpenAPI або RAML, а також включає такі можливості, як інтерактивна документація, генерація коду та інструменти управління API. Примітно, що Stoplight виділяється своїми сильними сторонами у візуальному проектуванні API та інтеграції з інструментами розробки, такими як GitHub та Jira. У згаданій статті автор розглядає проблему, що виникає під час створення та підтримки стандартів OpenAPI для тестування REST API.

Спеціальний інструмент під назвою Respector був представлений як перший метод використання статичного та символічного аналізу програм для генерації специфікацій для REST API з їхнього вихідного коду [34]. Надані експерименти показали, що Respector успішно виявив численні відсутні методи

кінцевих точок, параметри, обмеження та відповіді, а також виявив кілька розбіжностей між специфікаціями, наданими розробниками, та фактичними реалізаціями API. Більше того, Respector перевершив інші методи, які виводять специфікації з анотацій API або шляхом виклику API.

Зі зростанням популярності об'єктно-орієнтованих мов та портативності Java API, розробка та використання програмних компонентів, що можна повторно використовувати, стають все більш можливими [35]. Ефективність повторного використання компонентів значною мірою залежить від надійності цих компонентів, що досягається шляхом комплексного тестування.

Однак у літературі бракує практичних підходів для генерації вхідних даних та перевірки вихідних даних для численних необхідних тестових випадків. Автор представляє інструмент "Roast" та пов'язані з ним методи тестування Java API.

Практичність та ефективність цих методів демонструються за допомогою двох складних компонентів, з кількісними результатами, наданими для перевірки різних підходів.

У кожній з цих статей описані різні сильні сторони, будь то співпраця, інтеграція, інтерактивна документація чи управління API. Залежно від конкретних потреб астрономічного проекту щодо обробки астрономічних даних, одна з цих альтернатив може бути кращою, ніж Swagger, для реалізації специфікацій OpenAPI.

Основна частина

Розподілена інформаційна система, орієнтована на мікросервіси, для обробки астрономічних даних

Проектування сервісу HTTP API, пов'язаного з обробкою астрономічних даних, передбачає створення кінцевих точок, які дозволяють клієнтам взаємодіяти та отримувати дані про небесні об'єкти, астрономічні явища та іншу відповідну інформацію.

На діаграмі нижче представлено високорівневу архітектуру розробленої системи. Вона складається з кількох архітектурних компонентів, включаючи клієнтські програми, агрегатори API серверної частини та мікросервіси домену.

Мікросервіси забезпечують кілька каналів зв'язку, включаючи асинхронні та синхронні способи.

Синхронний канал зв'язку реалізовано шляхом надання HTTP API для зчитування моделі даних. Будь-яка модель даних виконується асинхронно через шину повідомлень (у цьому випадку RabbitMQ).

Оскільки отримання даних виконується через HTTP-запит, ми бачимо важливість інструменту OpenAPI та Swagger. Після оновлення даних через брокер повідомлень асинхронним способом, Swagger дозволяє отримати доступ до HTTP REST API синхронним способом, здійснюючи прямий HTTP-виклик до мікросервісу, який отримує доступ до сховища даних та перевіряє збережену в ньому інформацію.

Рисунок 1 ілюструє наведену діаграму як високорівневу архітектуру для системи, пов'язаної з астрономією, з використанням мікросервісів.

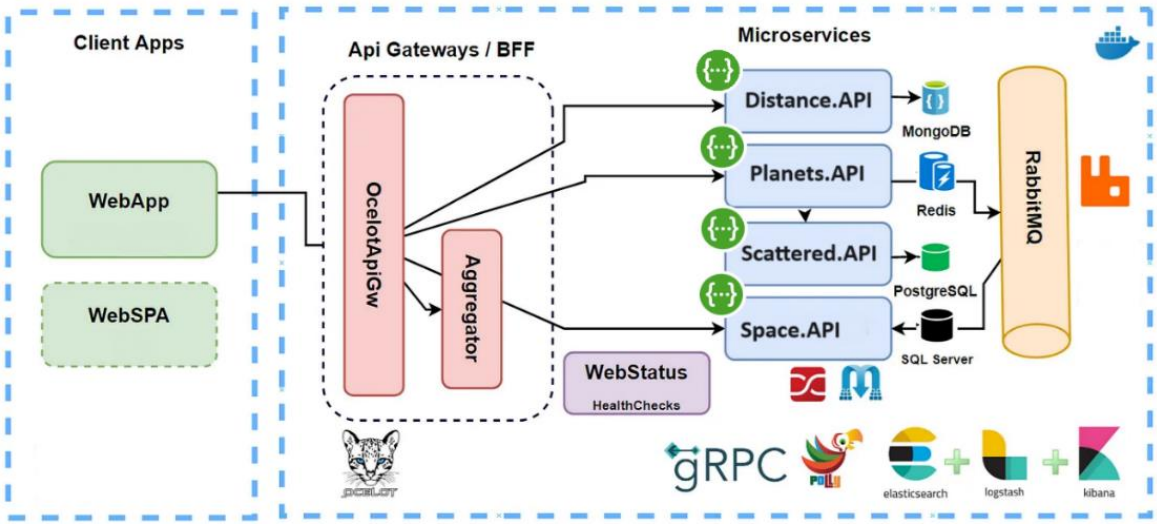

Рисунок 1 – Високорівнева архітектура для системи, пов'язаної з астрономією, з використанням мікросервісів

Ось детальний опис кожного компонента та їхньої взаємодії.

1. Клієнтські програми:

– **WebApp:** традиційний інтерфейс веб-програми, який взаємодіє зі шлюзом API;

– **WebSPA:** односторінковий додаток (SPA), який забезпечує більш динамічний користувацький досвід, а також взаємодіє зі шлюзом API.

2. Шлюзи API / BFF (Backend for Frontend):

– **OcelotApiGw:** це шлюз API, який обробляє запити від клієнтських програм та направляє їх до відповідних мікросервісів. Він надає такі функції, як автентифікація, авторизація, агрегація запитів тощо;

– **Aggregator:** цей компонент агрегує дані з кількох мікросервісів в одну відповідь, оптимізуючи кількість викликів, необхідних клієнтським програмам.

3. Мікросервіси:

– **Distance.API:** обробляє операції, пов'язані з астрономічними відстанями. Він використовує MongoDB для зберігання даних, забезпечуючи гнучке та масштабоване зберігання даних про відстані;

– **Planets.API:** керує даними, пов'язаними з планетами. Він використовує Redis, сховище даних в пам'яті, для підвищення швидкості доступу до даних та кешування;

– **Scattered.API:** ймовірно, має справу з розсіяними об'єктами в космосі, такими як астероїди або комети. Він використовує PostgreSQL, потужну реляційну базу даних з відкритим кодом;

– **Space.API:** керує загальними даними, пов'язаними з космосом. Він спирається на SQL Server, надійну реляційну систему баз даних від Microsoft.

4. RabbitMQ як брокер повідомлень, який використовується для асинхронного зв'язку між мікросервісами. Він забезпечує архітектуру, керовану

подіями, де сервіси можуть публікувати події та підписуватися на них без тісного зв'язку.

5. Додаткові компоненти:

а) **WebStatus (HealthChecks):** сервіс, який контролює стан справності різних мікросервісів, забезпечуючи їх оптимальну роботу. Він може надати інформацію про час безперебійної роботи та продуктивність сервісів;

б) **gRPC:** високопродуктивний фреймворк RPC з відкритим кодом, який можна використовувати для зв'язку між мікросервісами, пропонуючи такі переваги, як мовна агностицизм, низька затримка та ефективна серіалізація даних;

в) **Polly:** бібліотека стійкості .NET та обробки тимчасових помилок, яка дозволяє розробникам виражати такі політики, як Retry, Circuit Breaker, Timeout, Bulkhead Isolation та Fallback;

г) **ELK Stack (Elasticsearch, Logstash, Kibana):** набір інструментів для реєстрації, пошуку та візуалізації даних:

– **Elasticsearch:** пошукова та аналітична система;

– **Logstash:** конвеєр обробки даних, який отримує дані з кількох джерел, перетворює їх, а потім надсилає до сховища, такого як Elasticsearch;

– **Kibana:** інструмент візуалізації, що використовується для дослідження даних, що зберігаються в Elasticsearch, що забезпечує графічні представлення та панелі інструментів.

6. Потік даних:

– **Взаємодія з клієнтом:** користувачі взаємодіють з WebApp або WebSPA, який надсилає запити до OcelotApiGw;

– **Маршрутизація шлюзу API:** шлюз API направляє ці запити до відповідного мікросервісу (Distance.API, Planets.API, Scattered.API, Space.API);

– **Агрегація даних:** для складних запитів, що потребують даних з кількох джерел, агрегатор збирає необхідну інформацію;

– **Операції з базою даних:** кожен мікросервіс взаємодіє зі своєю відповідною базою даних (MongoDB, Redis, PostgreSQL, SQL Server) для виконання операцій CRUD;

– **Асинхронний зв'язок:** мікросервіси взаємодіють асинхронно через RabbitMQ, що дозволяє створювати масштабовану та роз'єднану архітектуру;

– **Моніторинг справності:** сервіс WebStatus постійно контролює справність усіх сервісів;

– **Журнал та візуалізація:** журнали та метрики збираються, обробляються та візуалізуються за допомогою стеку ELK, що полегшує моніторинг та налагодження.

Ця архітектура демонструє надійний та масштабований підхід до управління системою, пов'язаною з астрономією, за допомогою мікросервісів, API Gateway, асинхронного зв'язку та комплексних можливостей моніторингу та реєстрації стану. Вона використовує сучасні технології для забезпечення високої продуктивності, стійкості та зручності обслуговування. Як можна зазначити з

наведеної вище діаграми, мікросервіси, представлені HTTP-сервісами API, надають документацію OpenAPI, розкриваючи кінцеві точки Swagger.

Надана специфікація OpenAPI описує API для сервісу, пов'язаного з астрономією, з кількома кінцевими точками для керування та отримання даних про відстані, планети, розсіяні диски, космос та опорні зірки [36]. Нижче наведено детальний опис кожної частини специфікації.

Специфікація надається з використанням OpenAPI версії: 3.0.1. Специфікація реалізована за допомогою інструменту з відкритим кодом під назвою Swagger UI, і нижче наведено приклади використання цього інструменту API.

Перший розділ специфікації (/api/Distance) пов'язаний з вимірюванням астрономічної відстані. Існуючі кінцеві точки HTTP приймають HTTP GET та POST запити до сервісу, що дозволяє вводити запис щодо будь-якої відстані, а також отримувати вже існуючу інформацію.

Специфікацію OpenAPI можна знайти нижче.

GET: Отримує список відстаней.

Теги: Відстань.

Відповіді:

200: Успіх повертає масив об'єктів Distance у форматах text/plain, application/json або text/json.

POST: Створює новий запис відстані.

Теги: Відстань.

Тіло запиту: Приймає об'єкт Distance у форматах application/json, text/json або application/*+json.

Відповіді:

200: Успіх повертає створений об'єкт Distance.

Специфікація Swagger (/api/Planets) пов'язана з планетами у Всесвіті, надаючи кінцеві точки для доступу до всієї інформації, включаючи назви, порядок та планетарну систему. Існуючі контракти дозволяють отримувати існуючий список планет та записувати новий запис про планету, який нещодавно був відкритий.

Специфікацію OpenAPI можна знайти нижче.

GET: Отримує список планет.

Теги: Планети.

Відповіді:

200: Успіх повертає масив рядків, що представляють назви планет у форматах text/plain, application/json або text/json.

POST: Створює новий запис про планету.

Теги: Планети.

Тіло запиту: Приймає рядок у форматах application/json, text/json або application/*+json.

Відповіді:

200: Успіх.

Розсіяний диск – це віддалена область Сонячної системи, яка простягається за орбіту Нептуна. Вона населена групою малих крижаних тіл, відомих як об'єкти розсіяного диска (SDO). Ці об'єкти мають високоеліптичні орбіти, які віддаляють їх від Сонця в їхньому афелії (точці їхньої орбіти, найдальшій від Сонця) та ближче до Сонця в їхньому перигелії (точці їхньої орбіти, найближчій до Сонця). Ключові характеристики розсіяного диска включають орбітальні характеристики, походження, склад, відомі об'єкти.

Перелічені характеристики охоплені специфікацією OpenAPI (/api/ScatteredDisk), наведеною нижче.

GET: Отримує список об'єктів, пов'язаних з простором.

Теги: Пробіл.

Відповіді:

200: Успіх повертає масив рядків у форматах text/plain, application/json або text/json.

POST: Створює новий запис простору.

Теги: Пробіл.

Тіло запиту: Приймає рядок у форматах application/json, text/json або application/*+json.

Відповіді:

200: Успіх

Рисунок 2 ілюструє ефективність використання Swagger у контексті обробки астрономічних даних.

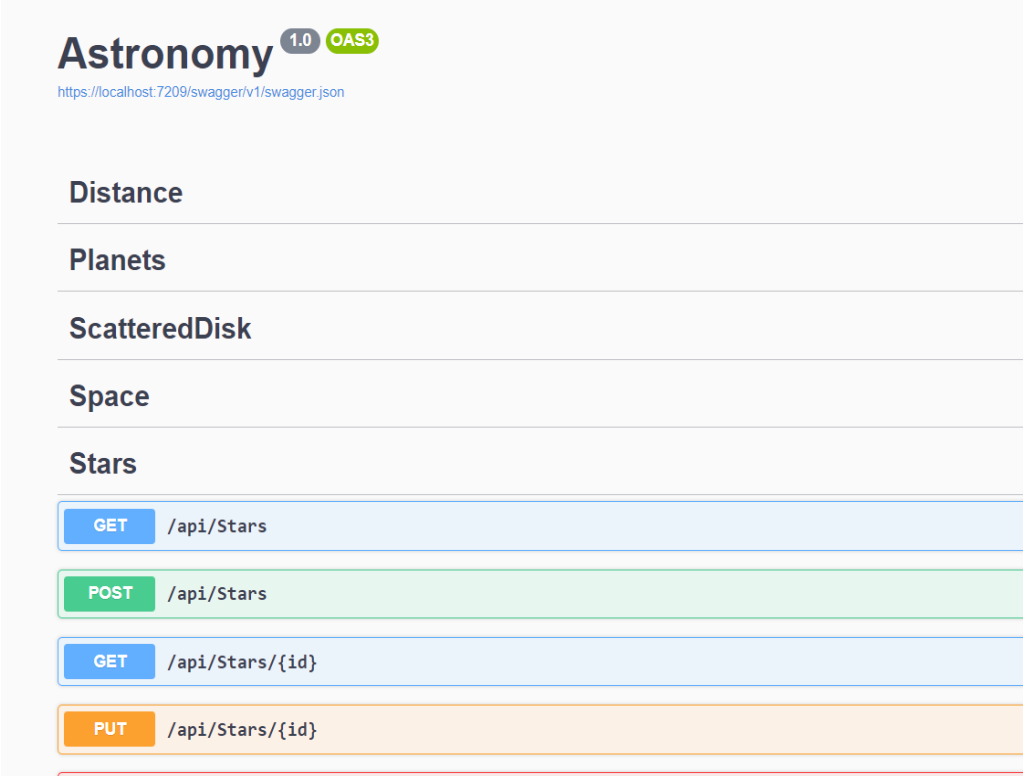

Рисунок 2 – Приклад перетворення специфікації OpenAPI JSON у зручний графічний інтерфейс користувача за допомогою інструменту Swagger

На зображенні нижче ми бачимо приклад перетворення специфікації OpenAPI JSON у зручний графічний інтерфейс користувача за допомогою інструменту Swagger.

Моделі даних визначають структуру ваших сутностей даних у C#. Для астрономічного API ці моделі представляють небесні об'єкти та їхні атрибути. Метою наступного архітектурного компонента є визначення моделі домену та основних атрибутів, необхідних під час обробки астрономічних даних. Наступні важливі атрибути повинні бути визначені всередині астрономічної моделі домену SSO [37]: маса, радіус, ідентифікатор (ім'я) тощо.

Екземпляри цих моделей використовуються у програмі для представлення та маніпулювання даними, пов'язаними із зірками. Візуальне представлення моделей баз даних підкреслює різні структури та технології, що використовуються для кожного мікросервісу. Наступна діаграма демонструє ці моделі (рис. 3).

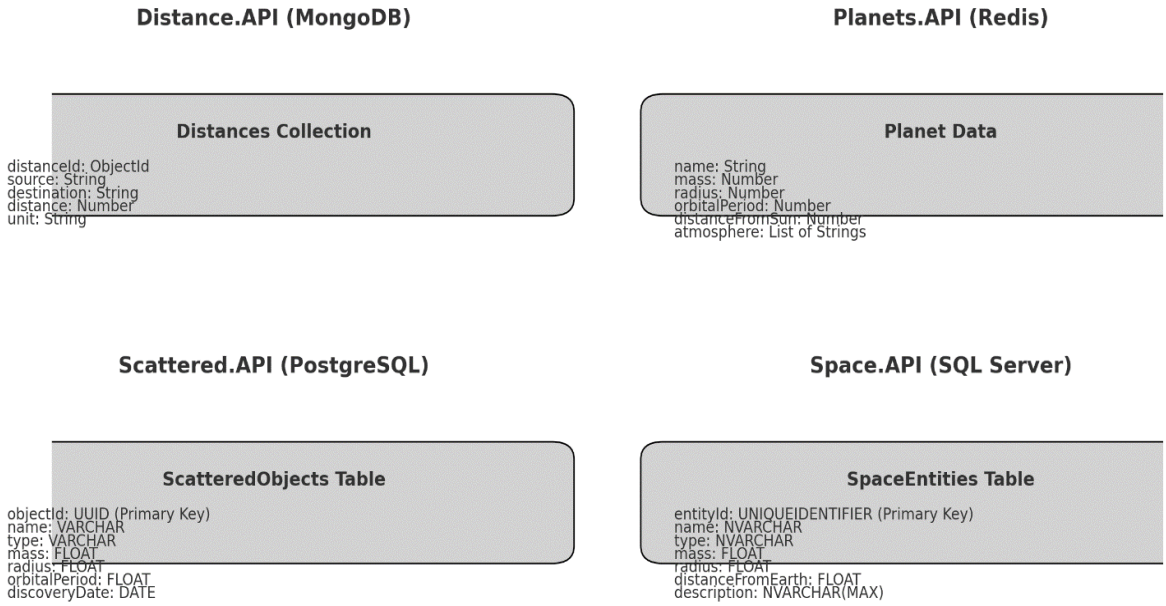

Рисунок 3 – Моделі даних для системи, пов'язаної з астрономією, з використанням мікросервісів

Distance.API (MongoDB). Колекція документів з різними полями для даних про відстані. Ця колекція зберігатиме інформацію про відстані між різними астрономічними об'єктами.

Кожен документ представлятиме певне вимірювання відстані, включаючи джерело та місце призначення вимірювання, значення відстані та одиницю вимірювання.

Модель відстані JSON представлена NoSQL (документно-орієнтованою) базою даних. Колекція бази даних називається Distances і включає наступний набір полів.

```
{
 "distanceId": "60c72b2f4f1a4e3d5c8b4567",
 "source": "Earth",
```

```
  "destination": "Mars",
  "distance": 0.52,
  "unit": "AU"
}
```

Planets.API (Redis). Структури даних в оперативній пам'яті для зберігання даних про планети. У Redis кожна планета зберігатиметься як хеш, де ключ – це унікальний ідентифікатор планети (наприклад, planet:1), а значення – це хеш, що містить різні атрибути планети, такі як назва, маса, радіус, орбітальний період, відстань від сонця та склад атмосфери.

Redis тут використовується для швидких операцій читання та запису, що корисно для часто використовуваних даних.

Модель JSON для планет представлена базою даних NoSQL (сховище ключ-значення). Оскільки це сховище ключ-значення, дані слід зберігати в одному рядку шляхом хешування або серіалізації в рядок JSON.

```
{
  "name": "Earth",
  "mass": 5.972e24,
  "radius": 6371,
  "orbitalPeriod": 365.25,
  "distanceFromSun": 1.00,
  "atmosphere": ["Nitrogen", "Oxygen", "Argon", "Carbon Dioxide"]
}
```

Scattered.API (PostgreSQL). Реляційна таблиця з полями для даних розсіяних об'єктів. Ця таблиця зберігатиме дані про розсіяні астрономічні об'єкти, такі як астероїди та комети.

Кожен рядок представляє об'єкт з атрибутами, включаючи його ідентифікатор, назву, тип, масу, радіус, орбітальний період та дату відкриття. PostgreSQL обрано за його відповідність ACID та потужні можливості запитів.

Розсіяні об'єкти зберігаються в реляційній базі даних SQL з унікальним ідентифікатором як первинним ключем для кожного розсіяного об'єкта та списком пов'язаних атрибутів.

```
CREATE TABLE ScatteredObjects (
  objectId UUID PRIMARY KEY,
  name VARCHAR(255),
  type VARCHAR(50),
  mass FLOAT,
  radius FLOAT,
  orbitalPeriod FLOAT,
  discoveryDate DATE
);
```

Space.API (SQL Server). Реляційна таблиця для даних космічних об'єктів з вичерпними полями для детальної інформації.

Ця таблиця зберігатиме загальну інформацію про різні космічні об'єкти, такі як зірки, галактики та туманності.

Кожен рядок представляє об'єкт з атрибутами, включаючи його ідентифікатор, назву, тип, масу, радіус, відстань від Землі та опис.

SQL Server використовується тут завдяки своїм корпоративним функціям та надійній продуктивності.

Модель Space SQL також представлена як реляційна таблиця бази даних з відповідним основним атрибутом та списком призначених атрибутів.

```
CREATE TABLE SpaceEntities (
  entityId UNIQUEIDENTIFIER PRIMARY KEY,
  name NVARCHAR(255),
  type NVARCHAR(50),
  mass FLOAT,
  radius FLOAT,
  distanceFromEarth FLOAT,
  description NVARCHAR(MAX)
);
```

Ця архітектура використовує сильні сторони кожної технології баз даних, забезпечуючи оптимальну продуктивність, масштабованість та гнучкість для обробки різноманітних вимог до даних в HTTP-сервісі, пов'язаному з астрономією.

Каркас запропонованої специфікації OpenAPI в розподіленій інформаційній системі, орієнтованій на мікросервіси, для обробки астрономічних даних був протестований в рамках програмного забезпечення Lemur проекту Collection Light Technology (CoLiTec) (https://colitec.space) [38].

Конкретні модулі та сервіси, пов'язані з математичними методами та алгоритмами в програмному забезпеченні Lemur:

– автоматичне калібрування кадру;

– косметична корекція кадру;

– функція відстеження та стекування;

– вирівнювання яскравості;

– вирівнювання фону [6];

– фільтрація астрономічних зображень [5, 27];

– визначення контурів об'єктів;

– розпізнавання зображень [30, 31];

– формування типової форми [19];

– виявлення рухомих об'єктів (з майже нульовим, нормальним, швидким видимим рухом) [39];

– повністю автоматизований робустний метод астрометричної редукції [7];

– повністю автоматизований робустний метод фотометричного відновлення [40];

– підтримка багатопотокової обробки;

– передача астрономічних даних з проміжним зберіганням.

Більш детальна інформація про програмне забезпечення Lemur проекту CoLiTec представлена в цих статтях [41, 42, 43] та дослідженнях [44, 45].

Приклад даних JSON, реалізованих в рамках розподіленої мікросервісно-орієнтованої інформаційної системи для обробки астрономічних даних для програмного забезпечення Lemur, є відповіддю на API, що надає інформацію про відстань між небесними тілами, згаданими нижче.


```
{
 "origin": {
  "name": "Earth",
  "type": "Planet",
  "coordinates": {
   "x": 0.0,
   "y": 0.0,
   "z": 0.0
  }
 },
 "destination": {
  "name": "Mars",
  "type": "Planet",
  "coordinates": {
   "x": 1.5,
   "y": 0.5,
   "z": 0.2
  }
 },
 "distance": {
  "unit": "AU",
  "value": 1.52
 },
 "travelTime": {
  "unit": "days",
  "value": 300
 },
 "path": [
  {
   "x": 0.5,
   "y": 0.2,
   "z": 0.1
  },
```

```
    {
      "x": 1.0,
      "y": 0.4,
      "z": 0.15
    },
    {
      "x": 1.5,
      "y": 0.5,
      "z": 0.2
    }
  ],
  "metadata": {
    "requestTime": "2026-05-18T12:34:56Z",
    "responseTime": "2026-05-18T12:34:57Z",
    "service": "DistanceAPI"
  }
}
```


Представлена структура JSON містить таку цінну інформацію:

– **origin**. Інформація про початкову точку розрахунку відстані, включаючи назву, тип (наприклад, планета, зірка) та координати у 3D-просторі;

– **destination**. Інформація про кінцеву точку розрахунку відстані, подібну до точки початку координат;

– **distance**. Розрахована відстань між точкою початку координат та точкою призначення, а також одиниця вимірювання (наприклад, астрономічні одиниці - AU);

– **travelTime**. Орієнтовний час подорожі для подолання відстані, а також одиниця вимірювання (наприклад, дні);

– **path**. Масив координат, що представляє шлях від точки початку до точки призначення;

– **metadata**. Додаткова інформація про запит API, включаючи час запиту та відповіді, а також назву сервісу, який надав дані.

Ця структура JSON розроблена як комплексна та може бути розширена на основі конкретних вимог та додаткових атрибутів, які можуть бути релевантними для API відстані в архітектурі мікросервісів.

Результати досліджень

Обговорення

Впровадження Swagger в архітектуру мікросервісів надає численні переваги, значно покращуючи як фази розробки, так і фази підтримки сервісно-орієнтованих програм. Swagger, як фреймворк з відкритим кодом, спрощує проектування, створення, документування та використання RESTful веб-сервісів.

Його здатність генерувати інтерактивну документацію API з анотацій коду гарантує, що всі зацікавлені сторони мають доступ до актуальної та точної інформації про API, сприяючи кращій комунікації та співпраці в командах розробників.

Однією з основних переваг використання Swagger є стандартизована документація, яку він надає. Ця стандартизація гарантує, що кожен мікросервіс дотримується узгодженого формату, спрощуючи розуміння та використання API в різних сервісах. Ця узгодженість є критично важливою в архітектурі мікросервісів, де кілька сервісів повинні безперешкодно взаємодіяти, а розробникам може знадобитися одночасно працювати з різними API.

Інтерактивна документація Swagger також дозволяє розробникам тестувати API безпосередньо з інтерфейсу документації. Ця функція спрощує процес розробки та налагодження, дозволяючи швидше виконувати ітерації та ефективніше виправляти несправності. Забезпечуючи чіткий та інтерактивний спосіб візуалізації та тестування кінцевих точок API, Swagger скорочує криву навчання для нових розробників та підвищує продуктивність.

Крім того, Swagger підтримує автоматичну генерацію коду для клієнтів API різними мовами програмування, що прискорює розробку клієнтських програм та сервісів. Ця автоматизація мінімізує помилки ручного кодування та гарантує, що клієнтські реалізації відповідають специфікаціям API, що ще більше сприяє узгодженості та надійності в усій системі.

В екосистемі мікросервісів, де сервіси часто розробляються, розгортаються та масштабуються незалежно, підтримка актуальної документації може бути складною. Swagger вирішує цю проблему, інтегруючись безпосередньо з кодовою базою, гарантуючи, що будь-які зміни в API автоматично відображаються в документації. Ця інтеграція особливо корисна для конвеєрів безперервної інтеграції та безперервного розгортання (CI/CD), гарантуючи, що зміни API документуються та тестуються протягом усього життєвого циклу розробки.

Використання Swagger також покращує видимість та зручність використання API. Завдяки добре документованому API внутрішні та зовнішні розробники можуть легко досліджувати доступні кінцеві точки, розуміти вимоги до вхідних та вихідних даних, а також ефективніше інтегрувати сервіси. Ця можливість виявлення є вирішальною для сприяння інноваціям та надання розробникам можливості використовувати існуючі сервіси для створення нових функцій.

Крім того, підтримка Swagger версій API гарантує, що зміни та оновлення API можна керувати без порушення роботи існуючих споживачів. Ця можливість є важливою в архітектурі мікросервісів, де різні сервіси та клієнти можуть залежати від різних версій одного й того ж API. Чітко документуючи кінцеві точки з версіями, Swagger допомагає підтримувати зворотну сумісність та плавні переходи під час оновлень.

Загалом, використання Swagger в архітектурі мікросервісів оптимізує робочі процеси розробки, покращує якість API та покращує співпрацю між

командами. Його вичерпна документація, інтерактивні функції та можливості автоматизації роблять його незамінним інструментом для управління складними екосистемами сервісів.

Забезпечуючи належну документацію, легкість тестування та послідовне впровадження API, Swagger сприяє надійності, масштабованості та зручності обслуговування програм на основі мікросервісів. На завершення, інтеграція Swagger в архітектуру мікросервісів є стратегічним рішенням, яке може значно покращити як досвід розробки, так і операційну ефективність сервісно-орієнтованих систем.

Надане JSON-представлення системи обробки астрономічних даних демонструє ефективний та структурований підхід до надання важливих астрономічних даних в рамках архітектури мікросервісів. Інкапсулюючи деталі джерела та призначення, включаючи їхні координати та типи, API пропонує точну та вичерпну інформацію про небесні тіла.

Включення вимірювань відстані в астрономічних одиницях (AU) гарантує, що дані є науково релевантними та придатними для використання в різних астрономічних розрахунках та аналізах. Крім того, розрахунковий час подорожі між небесними тілами, виражений у днях, надає цінну інформацію для планування гіпотетичних космічних місій або розуміння відносних відстаней у космосі.

Масив шляхів, який детально описує координати від джерела до місця призначення, додає ще один рівень корисності, потенційно допомагаючи у візуалізації подорожі космосом. Ця функція особливо корисна для освітніх інструментів, симуляцій та візуальних представлень у програмах, які потребують детальної траєкторії космічних подорожей.

Метадані, що охоплюють час запитів та відповідей, а також ідентифікацію сервісу, підвищують прозорість та відстежуваність API. Такі метадані мають вирішальне значення для моніторингу та налагодження в розподіленій системі, гарантуючи, що сервіс залишається надійним та зручним у обслуговуванні.

У контексті архітектури мікросервісів ця структура JSON ілюструє, як сервіси можуть бути модульними та незалежно масштабованими, що сприяє кращому використанню ресурсів та спрощенню управління. Використання чітко визначених кінцевих точок та чітких контрактів даних робить Distance API надійним компонентом, який може безперешкодно інтегруватися з іншими мікросервісами, такими як ті, що працюють з планетарною інформацією або об'єктами розсіяного диска.

Цей модульний підхід не тільки підвищує стійкість системи, але й дозволяє незалежні оновлення та масштабування окремих сервісів, покращуючи загальну продуктивність системи та зручність обслуговування.

Більше того, розширюваність структури JSON гарантує, що API може розвиватися разом зі зростаючими потребами своїх користувачів, враховуючи нові точки даних та функції, не порушуючи існуючу функціональність. Ця передбачливість у дизайні підтримує довгострокову стійкість та адаптивність сервісу. Детальна інформація, що надається Distance API, може служити різним

зацікавленим сторонам, від дослідників та викладачів до ентузіастів космосу та розробників, що робить його універсальним інструментом у галузі астрономії.

Структурована модель даних допомагає у створенні зручних інтерфейсів та інтуїтивно зрозумілих візуалізацій, сприяючи більшій залученості та розумінню серед користувачів. Дотримуючись найкращих практик у розробці API, таких як чіткі правила іменування та узгоджені формати даних, Distance API встановлює стандарт для інших сервісів в архітектурі мікросервісів. Така узгодженість забезпечує цілісний користувацький досвід та спрощує процес розробки нових функцій та сервісів.

Висновки

Ми представили використання специфікації OpenAPI в розподіленій мікросервісно-орієнтованій інформаційній системі для обробки астрономічних даних. Спільною метою всіх науково-технологічних алгоритмів і методів є автоматизація максимальної кількості доступних процесів без будь-яких дій людини.

У загальних випадках це може бути виконано за допомогою різних розподілених мікросервісно-орієнтованих інформаційних систем для обробки астрономічних даних. У цих конвеєрах різні завдання інтелектуального аналізу даних та пошуку знань у базах даних використовуються для пришвидшення та оптимізації обробки астрономічних даних.

Запропоноване використання специфікації OpenAPI в розподіленій мікросервісно-орієнтованій інформаційній системі для обробки астрономічних даних значно покращує сумісність, масштабованість та зручність обслуговування системи.

Розроблений скелет реального прикладу астрономічної системи обробки даних реалізовано за допомогою фреймворку .Net Core та мови програмування C#. Реалізація Swagger в мікросервісній архітектурі має численні переваги, значно покращуючи як фази розробки, так і фази підтримки сервісно-орієнтованих застосунків.

Розроблений скелет та запропонований підхід будуть корисними для різних мікросервісно-орієнтованих інформаційних систем для обробки астрономічних даних.

Його можна використовувати для всіх видів обробки астрономічних зображень, використовуючи різні математичні методи та алгоритми, реалізовані як інструмент, модуль або сервіс. Ще одним гарним прикладом застосування запропонованого скелета є реалізація концепції віртуальної обсерваторії (ВО).

Сучасні міжнародні астрономічні, астрометричні та фотометричні каталоги зараз доступні в хмарі, тому будь-яка взаємодія з такими даними з них вимагає інтеграції сервісів для обробки. Мікросервісно-орієнтована архітектура також буде дуже корисною в складних інформаційних системах для обробки

астрономічних даних з інтеграцією принципів безперервної інтеграції/безперервної доставки (CI/CD).

Подальші дослідження будуть проведені щодо тестування [46] та інтеграції запропонованої специфікації OpenAPI в розподілену мікросервісно-орієнтовану інформаційну систему для обробки астрономічних даних в рамках програмного забезпечення Lemur проекту Collection Light Technology (CoLiTec) [47].

Список літератури.

1. Wheeler, L., et al. (2024). Risk assessment for asteroid impact threat scenarios. Acta Astronautica, 216, 468-487. https://doi.org/10.1016/j.actaastro.2023.12.049.

2. Akhmetov, V., et al. (2019). Cloud computing analysis of Indian ASAT test on March 27, 2019. IEEE International Scientific-Practical Conference: Problems of Infocommunications Science and Technology. (p. 315-318). https://doi.org/10.1109/PICST47496.2019.9061243.

3. Savanevych, V., et al. (2023). Mathematical methods for an accurate navigation of the robotic telescopes. Mathematics, 11(10), 2246. https://doi.org/10.3390/math11102246.

4. Klette, R. (2014). Concise computer vision: An introduction into theory and algorithms. Springer.

5. Vlasenko, V., et al. (2024). Devising a procedure for the brightness alignment of astronomical frames background by a high frequency filtration to improve accuracy of the brightness estimation of objects. Eastern-European Journal of Enterprise Technologies, 2(2-128), 31-38. https://doi.org/10.15587/1729-4061.2024.301327.

6. Parimucha, Š., et al. (2019). CoLiTecVS – A new tool for an automated reduction of photometric observations. Contributions of the Astronomical Observatory Skalnaté Pleso, 49(2), 151-153.

7. Akhmetov, V., et al. (2020). Astrometric reduction of the wide-field images. Advances in Intelligent Systems and Computing. Vol. 1080. (p. 896-909). https://doi.org/10.1007/978-3-030-33695-0_58.

8. Kudak, V., Epishev, V., Perig, V., & Neybauer, I. (2017). Determining the orientation and spin period of TOPEX/Poseidon satellite by a photometric method. Astrophysical Bulletin, 72(3), 340-348. https://doi.org/10.1134/S1990341317030233.

9. Khlamov, S., & Savanevych, V. (2020). Big astronomical datasets and discovery of new celestial bodies in the Solar System in automated mode by the CoLiTec software. Knowledge discovery in big data from astronomy and earth observation: Astrogeoinformatics. (p. 331-345). https://doi.org/10.1016/B978-0-12-819154-5.00030-8.

10. Troianskyi, V., Godunova, V., Serebryanskiy, A., et al. (2024). Optical observations of the potentially hazardous asteroid (4660) Nereus at opposition 2021. Icarus, 420, 116146. https://doi.org/10.1016/j.icarus.2024.116146.

11. Adam, G., Kontaxis, P., Doulos, L., Madias, E.-N., Bouroussis, C., & Topalis, F. (2019). Embedded microcontroller with a CCD camera as a digital lighting control system. Electronics, 8(1). https://doi.org/10.3390/electronics8010033.

12. Oszkiewicz, D., et al. (2023). Spins and shapes of basaltic asteroids and the missing mantle problem. Icarus, 397, 115520. https://doi.org/10.1016/j.icarus.2023.115520.

13. Vavilova, I. B., Yatskiv, Y. S., Pakuliak, L. K., et al. (2016). UkrVO astroinformatics software and web-services. Proceedings of the International Astronomical Union, 12(S325), 361-366. https://doi.org/10.1017/S1743921317001661.

14. Akhmetov, V., et al. (2019). Fast coordinate cross-match tool for large astronomical catalogue. Advances in Intelligent Systems and Computing. Vol. 871. (p. 3-16). https://doi.org/10.1007/978-3-030-01069-0_1.

15. Borne, K. D. (2008). Scientific data mining in astronomy. Next generation of data mining. (p. 115-138). Chapman & Hall/CRC.

16. Zhang, Y., & Zhao, Y. (2015). Astronomy in the big data era. Data Science Journal, 14.

17. Raychev, N. (2020). Test automation in microservice architecture. IEEE Spectrum, 8(7).

18. Huang, R., Motwani, M., Martinez, I., & Orso, A. (2024). Generating REST API specifications through static analysis. Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE). (p. 1-13). https://doi.org/10.1145/3597503.3639137.

19. Savanevych, V., et al. (2022). Formation of a typical form of an object image in a series of digital frames. Eastern-European Journal of Enterprise Technologies, 6(2-120), 51-59. https://doi.org/10.15587/1729-4061.2022.266988.

20. Burger, W., & Burge, M. (2009). Principles of digital image processing: Fundamental techniques. Springer.

21. Khlamov, S., Tabakova, I., Trunova, T., & Deineko, Z. (2022). Machine vision for astronomical images using the Canny edge detector. CEUR Workshop Proceedings, 3384, 1-10.

22. Troianskyi, V., Kankiewicz, P., & Oszkiewicz, D. (2023). Dynamical evolution of basaltic asteroids outside the Vesta family in the inner main belt. Astronomy & Astrophysics, 672, A97. https://doi.org/10.1051/0004-6361/202245678.

23. Oszkiewicz, D., et al. (2020). Spin rates of V-type asteroids. Astronomy & Astrophysics, 643, A117. https://doi.org/10.1051/0004-6361/202038062.

24. Savanevych, V., et al. (2022). CoLiTecVS software for the automated reduction of photometric observations in CCD-frames. Astronomy and Computing, 40, 100605. https://doi.org/10.1016/j.ascom.2022.100605.

25. Khlamov, S., Savanevych, V., Vlasenko, V., Briukhovetskyi, O., Trunova, T., Levykin, I., Shvedun, V., & Tabakova, I. (2023). Development of the matched filtration of a blurred digital image using its typical form. Eastern-European Journal of Enterprise Technologies, 1(9-121), 62-71. https://doi.org/10.15587/1729-4061.2023.273674.

26. Khlamov, S., et al. (2022). Development of computational method for matched filtration with analytic profile of the blurred digital image. Eastern-European Journal of Enterprise Technologies, 5(4-119), 24-32. https://doi.org/10.15587/1729-4061.2022.265309.

27. Politsch, C. A., et al. (2020). Trend filtering–I. A modern statistical tool for time-domain astronomy and astronomical spectroscopy. Monthly Notices of the Royal Astronomical Society, 492(3), 4005-4018. https://doi.org/10.1093/mnras/staa106.

28. Dadkhah, M., et al. (2019). Methodology of wavelet analysis in research of dynamics of phishing attacks. International Journal of Advanced Intelligence Paradigms, 12(3-4), 220-238. https://doi.org/10.1504/IJAIP.2019.098561.

29. Kirichenko, L., Alghawli, A. S. A., & Radivilova, T. (2020). Generalized approach to analysis of multifractal properties from short time series. International Journal of Advanced Computer Science and Applications, 11(5), 183-198. https://doi.org/10.14569/IJACSA.2020.0110527.

30. Khlamov, S., et al. (2022). The astronomical object recognition and its near-zero motion detection in series of images by in situ modeling. Proceedings of the 29th IEEE International Conference on Systems, Signals, and Image Processing (IWSSIP 2022). (p. 1-4). https://doi.org/10.1109/IWSSIP55020.2022.9854475.

31. Khlamov, S., Tabakova, I., & Trunova, T. (2022). Recognition of the astronomical images using the Sobel filter. 29th IEEE International Conference on Systems, Signals, and Image Processing (IWSSIP 2022) (p. 1-4). https://doi.org/10.1109/IWSSIP55020.2022.9854425.

32. da Costa, D.N.N. (2022). Guidelines for testing microservice-based applications (Master's dissertation, 80 p.).

33. Stoplight. (n.d.). Solutions. Retrieved from https://stoplight.io/solutions.

34. Golmohammadi, A., Zhang, M., & Arcuri, A. (2023). Testing RESTful APIs: A survey. ACM Transactions on Software Engineering and Methodology, 33(1), 1-41.

35. Hoffman, D., & Strooper, P. (2002). Tools and techniques for Java API testing. Australian Software Engineering Conference. (p. 235-245). https://doi.org/10.1109/ASWEC.2000.844580.

36. Khlamov, S., et al. (2024). Automated data mining of the reference stars from astronomical CCD frames. CEUR Workshop Proceedings, 3668, 83-97.

37. Troianskyi, V., Kashuba, V., Bazyey, O., et al. (2023). First reported observation of asteroids 2017 AB8, 2017 QX33, and 2017 RV12. Contributions of the Astronomical Observatory Skalnaté Pleso, 53, 5-15. https://doi.org/10.31577/caosp.2023.53.2.5.

38. Khlamov, S., et al. (2024). Machine vision for astronomical images using the modern image processing algorithms implemented in the CoLiTec software. In Measurements and Instrumentation for Machine Vision. Chap. 12. (p. 269-310). CRC Press, Taylor & Francis Group. https://doi.org/10.1201/9781003343783-12.

39. Khlamov, S., et al. (2016). Development of computational method for detection of the object's near-zero apparent motion on the series of CCD-frames. Eastern-European Journal of Enterprise Technologies, 2(9(80)), 41-48. https://doi.org/10.15587/1729-4061.2016.65999.

40. Kudzej, I., et al. (2019). CoLiTecVS – A new tool for the automated reduction of photometric observations. Astronomische Nachrichten, 340(1-3), 68-70. https://doi.org/10.1002/asna.201913562.

41. Savanevych, V., et al. (2015). A new method based on the subpixel Gaussian model for accurate estimation of asteroid coordinates. Monthly Notices of the Royal Astronomical Society, 451(3), 3287-3298. https://doi.org/10.1093/mnras/stv1124.

42. Mykhailova, L., et al. (2014). Method of maximum likelihood estimation of compact group objects location on CCD-frame. Eastern-European Journal of Enterprise Technologies, 5(4), 16-22. https://doi.org/10.15587/1729-4061.2014.28028.

43. Savanevych, V., et al. (2018). A method of immediate detection of objects with a near-zero apparent motion in series of CCD-frames. Astronomy & Astrophysics, 609, A54. https://doi.org/10.1051/0004-6361/201630323.

44. Shvedun, V., et al. (2016). Statistical modelling for determination of perspective number of advertising legislation violations. Actual Problems of Economics, 184(10), 389-396.

45. Savanevych, V., et al. (2015). Comparative analysis of the positional accuracy of CCD measurements of small bodies in the solar system software CoLiTec and Astrometrica. Kinematics and Physics of Celestial Bodies, 31(6), 302-313.

46. Khlamov, S., Mendielieva, M., Vovk, O., Trunova, T., & Teslenko, Y. (2025). Performance percentile analysis for API-based testing. Information control systems and intelligent technologies. Achievements and applications. Section 2 “Intelligent systems and data analysis”. (p. 235-253). Liha-Pres. https://doi.org/10.36059/978-966-397-538-2-13.

47. Khlamov, S., et al. (2022). Data mining of the astronomical images by the CoLiTec software. CEUR Workshop Proceedings, 3171, 1043-1055.